

The Adaptive Inference Reader

Xinference Research

An annotated reading list of fourteen papers · 2026

Table of contents

One thesis, fourteen proofs	3
Scheduling Foundations	4
Memory That Adapts to the Request	5
Spending Compute Where the Token Is Hard	6
Precision as a Measured Decision	7
Serving Phases on Their Own Terms	8
Closing note	9

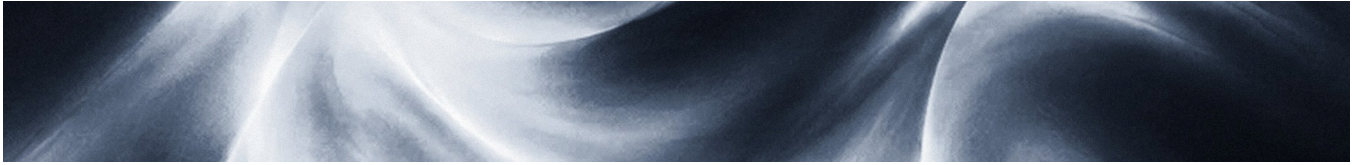
One thesis, fourteen proofs.

The research literature on efficient LLM inference looks, at first glance, like a pile of unrelated tricks: paged memory here, speculative tokens there, four-bit weights somewhere else. Read together, the papers tell one story. Every major efficiency gain of the past four years came from letting the serving system make decisions at runtime that used to be fixed at configuration time.

Memory that is allocated as sequences actually grow, not as they might grow. Compute that is spent where the token is hard, not uniformly. Prefill and decode phases scheduled by their real resource appetites, not by static worker pools. Quantization chosen by measured activation importance, not by a blanket bit-width. In every case the adaptive system beats the pre-configured one, and usually by enough to change the economics of a deployment.

This reader collects fourteen foundational papers behind that shift, in five sections: scheduling foundations, memory management, speculative execution, quantization, and disaggregated serving. Each entry gives the full citation and a short editorial note on what the paper showed and why it matters for production inference. Every citation has been verified against arXiv; the identifiers are printed so you can retrieve the originals directly.

None of these ideas is ours. The engineering judgment about which of them matter in production, and the conviction that adaptivity is the through-line, is the perspective we build Xinference on.



Section One

Scheduling Foundations

Paper 01 Xorbits: Automating Operator Tiling for Distributed Data Science

Weizheng Lu, Kaisheng He, Xuye Qin, et al. arXiv 2401.00865 (ICDE 2024).

The pre-history of this reader, and the origin of the team behind Xinference. Xorbits showed that the tiling of dataframe operators across a cluster, classically a decision made ahead of execution, could be automated and re-decided while the job runs, adapting partition sizes to the data actually flowing through. The paper's core finding generalizes far beyond dataframes: static partitioning leaves large factors of performance on the table whenever workloads are skewed or dynamic, which in production is always. Inference serving is exactly such a workload, and the same design instinct, defer the decision until the system can observe reality, runs through every paper that follows.

Paper 02 FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, Christopher Ré. arXiv 2205.14135 (2022).

The paper that made the memory hierarchy a first-class citizen of attention. FlashAttention computes exact attention while tiling reads and writes to fit GPU on-chip memory, recognizing that the bottleneck is data movement, not arithmetic. It reframed kernel design around what the hardware is actually doing during a forward pass, and nearly every serving stack in this reader assumes its lesson: performance comes from matching computation to observed hardware behavior, not from theoretical FLOP counts.

Memory That Adapts to the Request

Paper 03 Efficient Memory Management for Large Language Model Serving with PagedAttention

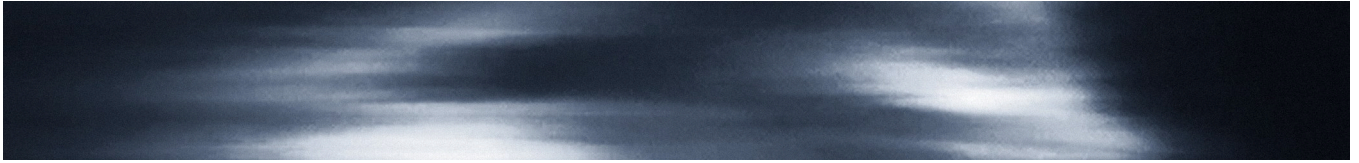
Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. arXiv 2309.06180 (SOSP 2023).

The vLLM paper, and arguably the single most consequential systems idea in modern LLM serving. Before it, servers reserved KV-cache memory for the longest possible sequence; most of that reservation sat idle. PagedAttention borrows virtual-memory paging from operating systems: cache blocks are allocated on demand as sequences actually grow. Fragmentation collapses, batch sizes rise, and throughput follows. It is the cleanest demonstration of the reader's thesis: replace a worst-case static reservation with runtime allocation and the same hardware serves multiples of the traffic.

Paper 04 SGLang: Efficient Execution of Structured Language Model Programs

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, et al. arXiv 2312.07104 (2023).

Where PagedAttention adapts memory to sequence length, SGLang adapts it to content. Its RadixAttention keeps a radix tree over KV caches so that requests sharing prefixes, system prompts, few-shot examples, agent scaffolding, reuse computation automatically at runtime. No operator declares what is shared; the tree discovers it. For agentic and template-heavy enterprise workloads, where prompts overlap massively, cache-aware runtime reuse is often worth more than a faster kernel.



Section Three

Spending Compute Where the Token Is Hard

Paper 05 Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan, Matan Kalman, Yossi Matias. arXiv 2211.17192 (ICML 2023).

The founding paper of speculative execution for LLMs. A small draft model proposes several tokens; the large model verifies them in one parallel pass; accepted tokens are free speedup, rejected ones cost nothing in output quality because the sampling distribution is provably preserved. The deep insight is that token difficulty varies wildly, and a static one-token-per-forward-pass loop ignores that variance. Speculation makes the decode loop adaptive to the text itself.

Paper 06 Accelerating Large Language Model Decoding with Speculative Sampling

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, et al. arXiv 2302.01318 (2023).

Independent, near-simultaneous discovery of the same mechanism, with a modified rejection-sampling scheme and results on a 70B-parameter production model. Worth reading alongside Paper 05 for its distributed-serving perspective: speedups survive contact with multi-accelerator sharding. When two groups arrive at one idea independently, the idea is usually load-bearing.

Paper 07 Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads

Tianle Cai, Yuhong Li, Zhengyang Geng, et al. arXiv 2401.10774 (2024).

Speculation without the second model: Medusa adds lightweight decoding heads to the model itself, drafting multiple future tokens and verifying them through tree attention. Removing the draft-model dependency makes speculative execution far easier to operate, one model to deploy, one to monitor, which matters in enterprise settings where every extra moving part is an operational liability.

Paper 08 Confident Adaptive Language Modeling

Tal Schuster, Adam Fisch, Jai Gupta, et al. arXiv 2207.07061 (NeurIPS 2022).

CALM asks the adaptive question inside the network: why should an easy token pay for all the layers a hard token needs? It exits computation early when intermediate confidence is high, with calibrated guarantees on output quality. Early-exit inference remains less deployed than speculation, but the paper states the principle of this whole section in its purest form: per-token compute should be a runtime decision.



Section Four

Precision as a Measured Decision

Paper 09 **GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers**

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, Dan Alistarh. arXiv 2210.17323 (ICLR 2023).

The paper that made 3- and 4-bit weights respectable. GPTQ quantizes one layer at a time using approximate second-order information, choosing each weight's rounding by its measured effect on the layer's output error rather than rounding uniformly. It put large open-weight models on hardware that enterprises actually own, and established the pattern for everything after it: quantization quality comes from measuring sensitivity, not from picking a bit-width.

Paper 10 **AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration**

Ji Lin, Jiaming Tang, Haotian Tang, et al. arXiv 2306.00978 (MLSys 2024).

AWQ's observation is disarmingly simple: a small fraction of weight channels does most of the work, and you can find them by watching activations, the model's runtime behavior, rather than the weights themselves. Protecting that salient fraction preserves accuracy at aggressive bit-widths without any backpropagation. It is the quantization chapter's version of the book's thesis: observe the system running, then decide.

Serving Phases on Their Own Terms

-
- Paper 11** **SARATHI: Efficient LLM Inference by Piggybacking Decodes with Chunked Prefills**
Amey Agrawal, Ashish Panwar, Jayashree Mohan, et al. arXiv 2308.16369 (2023).
- Prefill saturates compute; decode starves it. SARATHI chunks long prefills and piggybacks decode steps into the same batches, so every forward pass carries a balanced mix. It is runtime workload shaping: the scheduler composes each batch from the requests actually in flight, and both phases run closer to hardware limits.
-
- Paper 12** **Splitwise: Efficient Generative LLM Inference Using Phase Splitting**
Pratyush Patel, Esha Choukse, Chaojie Zhang, et al. arXiv 2311.18677 (ISCA 2024).
- If the phases differ that much, why serve them on the same machines? Splitwise routes prefill and decode to different hardware pools, sized and even provisioned with different GPU generations to match each phase's profile, with KV caches streamed between them. The cluster layout itself becomes an adaptive decision driven by measured phase characteristics, with direct consequences for cost and power at datacenter scale.
-
- Paper 13** **DistServe: Disaggregating Prefill and Decoding for Goodput-Optimized Large Language Model Serving**
Yinmin Zhong, Shengyu Liu, Junda Chen, et al. arXiv 2401.09670 (OSDI 2024).
- DistServe gives disaggregation its economics: optimize goodput, requests served within latency targets, rather than raw throughput. Co-locating the phases forces one compromise batching policy; separating them lets each phase meet its own service-level objective. For enterprises that buy inference by the SLO, this is the paper that connects scheduling architecture to what the business actually pays for.
-
- Paper 14** **Mixture-of-Depths: Dynamically Allocating Compute in Transformer-Based Language Models**
David Raposo, Sam Ritter, Blake Richards, et al. arXiv 2404.02258 (2024).
- The closing paper points at the frontier: adaptivity moving into the architecture itself. Mixture-of-Depths lets a transformer route tokens past blocks under a fixed compute budget, learning during training which tokens deserve depth. The boundary between model and serving system blurs; the network itself becomes the scheduler. Whatever serving stacks look like in five years, this direction of travel, decisions deferred to runtime, budgets enforced dynamically, is the safest prediction in the field.

Adaptivity is a property you deploy, not a paper you cite

The fourteen papers in this reader share one conclusion: systems that adapt while they run beat systems that are configured before they start. Xinference is our attempt to make that conclusion operational, an inference platform that supports more than 300 individual models through one API, applying these scheduling, memory, and quantization techniques inside your own infrastructure, under your own control.

All citations were verified against arXiv at the time of publication. We encourage you to read the originals; each identifier above retrieves the paper at arxiv.org.

Learn more [xinference.co](https://x inference.co)

Request demo xinference.co/xinference/demo.html

To learn more, visit xinference.co. For ongoing engineering analysis, follow Xinference on LinkedIn.

